ALGORITMOS Y ESTRUCTURAS DE DATOS
Laboratorio 8

Se busca realizar ejercicios de conceptos asociadas a programación, manejo de estructuras no lineales e interacciones con el sistema de archivos además de la salida y entrada estándar.

Objetivos

- Realizar los ejercicios utilizando el lenguaje de **programación C++.**
- Realizar una correcta creación y manipulación de estructuras.
- Realizar una correcta implementación de algoritmos que involucren decisiones y bucles.
- Realizar una correcta implementación de estructura de datos no lineales.

Nomenclatura para nombre de archivos fuentes

El nombre del archivo en el cual se almacena el código fuente debe considerar el siguiente formato: laboratorioN.EXT donde; EXT es la extensión del lenguaje de programación utilizado y N el número del Laboratorio.

Ayuda memoria

Ejercicio número 1

Crear una clase llamada Rectangle, considerando dos atributos privados y los métodos públicos. Incluir además un método para calcular el área.

Ejercicio número 2

Crear una clase llamada Point, considerando dos atributos privados y los métodos públicos para su definición y acceso. Implementar el uso de esta clase mediante la definición de objetivos a través del constructor y utilizando los métodos.

Ejercicio número 3

Crear una clase llamada LinkedList, considerando que contenga un atributo privado que apunte al primer nodo de la lista y los métodos públicos para insertar, buscar, eliminar e imprimir. Implementar el uso de las clases y pruebas de todas las funcionalidades.

Ejercicio número 4

Crear una clase llamada BinaryTree, considerando que contenga un atributo privado que apunte a la raíz del árbol y los métodos públicos para insertar, buscar e imprimir. Implementar el uso de las clases y pruebas de todas las funcionalidades.

Ejercicio número 5

Crear una clase llamada GeneralTree, considerando que contenga un atributo privado que apunte a la raíz del árbol y los métodos públicos para insertar, buscar e imprimir. Implementar el uso de las clases y pruebas de todas las funcionalidades.

Ayuda memoria

```cpp
#include <iostream>
#include <vector>

using namespace std;

// Class declaration
class MyClass {
public:
    MyClass(); // Default constructor
    MyClass(int num); // Constructor with parameter
    ~MyClass(); // Destructor

    void myMethod(); // Method declaration
    void setNum(int num); // Setter method
    int getNum(); // Getter method

private:
    int num_; // Private member variable
};

// Default constructor definition
MyClass::MyClass() {
    num_ = 0;
    cout << "Default constructor called." << endl;
}

// Constructor with parameter definition
MyClass::MyClass(int num) {
    num_ = num;
    cout << "Constructor with parameter called." << endl;
}

// Destructor definition
MyClass::~MyClass() {
    cout << "Destructor called." << endl;
}

// Method definition
void MyClass::myMethod() {
    cout << "myMethod() called." << endl;
}

// Setter method definition
void MyClass::setNum(int num) {
    num_ = num;
```

```
}

// Getter method definition
int MyClass::getNum() {
    return num_;
}

int main() {
    // Object creation using default constructor
    MyClass obj1;

    // Object creation using constructor with parameter
    MyClass obj2(10);

    // Method call
    obj1.myMethod();

    // Setter method call
    obj1.setNum(5);

    // Getter method call
    cout << "num_ value of obj1: " << obj1.getNum() << endl;

    // Vector of objects
    vector<MyClass> vec;
    vec.push_back(obj1);
    vec.push_back(obj2);

    return 0;
}
```